IEICE TRANS. FUNDAMENTALS, VOL.E82–A, NO.8 AUGUST 1999

PAPER *Special Section on Digital Signal Processing*

An Effective Architecture of the Pipelined LMS Adaptive Filters

Tadaaki KIMIJIMA[†], ***Student Member***, **Kiyoshi NISHIKAWA**[†], ***and*** **Hitoshi KIYA**[†], ***Members***

SUMMARY In this paper we propose a new pipelined architecture for the LMS adaptive filter which can be implemented with less than half the amount of calculation needed for the conventional architectures. Even though the proposed architecture reduces the required calculation, it can simultaneously produce good convergence characteristics, a short latency and high throughput characteristics.

key words: *adaptive filter, pipeline process, look-ahead transformantion*

1. Introduction

Researches on the development of the pipeline processing of gradient-type adaptive digital filters (ADFs) have been based on the delayed least mean square (delayed LMS: DLMS) algorithm [1]–[3]. The DLMS is a modified version of the LMS that is suited for pipelined processing. It has D units of delay in its error feedback path: pipelining of the DLMS is accomplished by moving these delays and placing them at every tap of the filter [1]–[3]. However, the DLMS has the disadvantage that the convergence characteristics worsen as the amount of delay increases even though it is necessary to increase the amount of delay; the finest grain pipelining is realized when D is set to be the length of the ADF. In addition, the latency of the architecture increases in proportion to the filter length [4]. We should consider these two issues when implementing ADFs using the pipeline technique.

To improve the convergence characteristics, several architectures have been proposed [4]–[6] so far. These architectures are based on the equivalent transformation reported shown in [7]. Improvement of convergence is achieved by adding a correctional term to the DLMS which transforms the DLMS into the LMS. Although this addition of the correctional term improves the convergence, it requires a large amount of calculations to be implemented. Moreover, these architectures cannot reduce the amount of output latency because they are based on the DLMS [4].

Recently an architecture that is based on the LMS was proposed [8]. This achieves convergence characteristics identical to those of the LMS without producing the output latency. Even if we use this architecture however, the problem of a large amount of calculators will remain since the equivalent expression of the LMS, which was introduced in [8] to achieve pipelining, requires about the same large amount of calculations used in the methods of [4]–[6].

In this paper we propose an effective architecture of the pipelined LMS ADFs. The main principle of the proposed architecture is the use of binary-tree adder in calculating the output signal and the insertion of the minimum amount of delay to shorten the critical path. As a result, the proposed architecture enables us to simultaneously produce good convergence characteristics, a short latency, and high throughput characteristics with less than half the amount of calculation needed for the conventional architectures.

This paper is organized as follows. In Sect. 2, we summarize the conventional architectures. Then, we give the derivation of the proposed architecture in Sect. 3. In Sect. 4, a comparison of the characteristics of the proposed architecture with the conventional ones is shown. The conclusion is given in Sect. 5.

2. Conventional Methods

First, we briefly describe the conventional methods and point out their problems.

2.1 The DLMS Algorithm

Let us start with a review of the DLMS algorithm [1]–[3]. The DLMS has D $(0 \leq D \leq N)$ units of delay in the error feedback path. By moving these delays and placing them at every tap of the filter, pipelining of the DLMS is accomplished.

By assuming that the ADF is an FIR filter, whose impulse response is denoted by $w_k(n)$, we can express the output signal $y(n)$ at time n as

$$y(n) = \sum_{k=0}^{N-1} x(n-k)w_k(n), \tag{1}$$

where $x(n)$ and N are, respectively, the input signal and the filter length. The DLMS is described by

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu e(n-D)\mathbf{x}(n-D) \tag{2}$$

$$e(n-D) = d(n-D) - \mathbf{W}^T(n-D)\mathbf{x}(n-D), \tag{3}$$

Manuscript received December 3, 1998.
Manuscript revised March 2, 1999.

[†]The authors are with the Department of Electronics Engineering, Graduate School of Engineering, Tokyo Metropolitan University, Hachioji-shi, 192–0397 Japan.

where $e(n-D)$ is the error signal of the DLMS and $d(n)$ is the desired signal; $\mathbf{W}(n)$ and $\mathbf{x}(n)$ are, respectively, the filter coefficient vector and the input vector, and they are given by

$$\mathbf{W}(n) = [w_0(n), w_1(n) \cdots w_{N-1}(n)]^T \quad (4)$$
$$\mathbf{x}(n) = [x(n), x(n-1) \cdots x(n-N+1)]^T, \quad (5)$$

where $[\mathbf{x}]^T$ indicates the transpose of a vector $\mathbf{x}$. Figure 1 shows the block diagram of the DLMS.

When the DLMS is used to pipeline ADFs, higher throughput can be obtained by increasing the amount of delay D. At the same time however, convergence characteristics become poor as the amount of delay increases. This degradation of the convergence is caused by the time lag between $\mathbf{W}(n)$ in Eq. (2) and $\mathbf{W}(n-D)$ in Eq. (3), and the time lag becomes large as the filter length increases. Hence, as the filter length increases, the convergence characteristics become poorer [5], [6].

2.2 Architectures Based on the Transformation Shown in [7]

To improve the convergence characteristics, several architectures were proposed [4]–[6]. These architectures are based on the transformation of the DLMS into the LMS, which was reported in [7]. In these architectures, the convergence characteristics are improved by adding a correctional term to the error signal of the DLMS.

Since the proposed architecture will be derived by using this transformation, we will give a brief explanation of the transformation. By applying *the look-ahead transformation* [4]–[6] to Eq. (2) for $\delta - 1$ $(1 \le \delta \le D)$ times, we express $\mathbf{W}(n+1)$ as

$$\mathbf{W}(n+1) = \mathbf{W}(n-\delta+1) + \sum_{i=0}^{\delta-1} \mu e(n-D-i)\mathbf{x}(n-D-i), \quad (6)$$

and by shifting this equation D times and rearranging it, we get

$$\mathbf{W}(n-D) = \mathbf{W}(n-D+\delta) + \sum_{i=0}^{\delta-1} \mu e(n-2D+\delta-i-1) \cdot \mathbf{x}(n-2D+\delta-i-1). \quad (7)$$

The following equation is obtained by substituting Eq. (7) into Eq. (3):

$$\begin{aligned} e(n-D) &= d(n-D) - \mathbf{W}^T(n-D)\mathbf{x}(n-D) \\ &= d(n-D) - \mathbf{W}^T(n-D+\delta)\mathbf{x}(n-D) \\ &\quad + \mu \sum_{i=0}^{\delta-1} e(n-2D+i)\mathbf{x}^T(n-2D+i) \\ &\quad \cdot \mathbf{x}(n-D). \end{aligned} \quad (8)$$

We can rewrite Eq. (8) as

$$\varepsilon(n-D) = e(n-D) - \Lambda(n), \quad (9)$$

where $\varepsilon(n-D)$ and $\Lambda(n)$ are defined as

$$\varepsilon(n-D) = d(n-D) - \mathbf{W}^T(n-D+\delta)\mathbf{x}(n-D) \quad (10)$$

$$\Lambda(n) = \mu \sum_{i=1}^{D-1} e(n-2D+i)\mathbf{x}(n-2D+i)^T \cdot \mathbf{x}(n-D). \quad (11)$$

By using $\varepsilon(n-D)$ instead of $e(n-D)$, we modify the DLMS as

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu\varepsilon(n-D)\mathbf{x}(n-D) \quad (12)$$

$$\varepsilon(n-D) = d(n-D) - \mathbf{W}^T(n-D)\mathbf{x}(n-D) - \Lambda(n) \quad (13)$$

$$\Lambda(n) = \mu \sum_{i=1}^{D-1} e(n-2D+i)\mathbf{x}(n-2D+i)^T \cdot \mathbf{x}(n-D), \quad (14)$$

where $\Lambda(n)$ is the correctional term [5], [6]. By adding the correctional term $\Lambda(n)$ to the error signal of the DLMS, we can decrease the time lag between $\mathbf{W}(n)$ in Eq. (12) and $\mathbf{W}(n-D+\delta)$ in Eq. (10) without reducing the inserted delays D [4]–[6]. Figure 2 shows the block diagram of the modified DLMS architectures. Note that when $\delta = 0$, this algorithm is reduced to the DLMS. When $\delta = D$, this algorithm becomes the LMS, which uses $x(n-D)$ and $d(n-D)$ instead of $x(n)$ and $d(n)$ in calculating error signal.

Although these approaches improve the convergence, a large amount of calculation is needed to implement the architectures: as is shown in Eq. (14), the amount of calculation needed to implement $\Lambda(n)$ depends on D. Delays, however, are required in order to

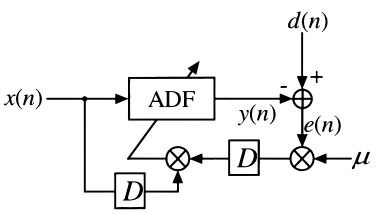

Fig. 1 Block diagram of the DLMS.

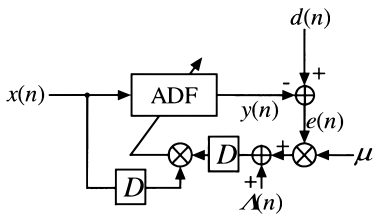

Fig. 2 Block diagram of architectures based on the transformation shown in [7].

maintain the high throughput characteristics. Moreover, these architectures cannot reduce the output latency [4], where we use the term latency as required clocks to calculate the output signal after input the signal.

2.3 Architecture Based on the LMS

In addition to the previously mentioned architectures, an architecture that is based on the LMS was also proposed [8]. This architecture shows that the LMS can be pipelined by using an equivalent expression of the LMS even though this had been considered to be impossible. This architecture has convergence characteristics and latency identical to those of the LMS. But it has the same problem as the architectures mentioned in Sect. 2.2: it requires a large amount of calculation in order to be implemented.

3. Proposed Architecture

Here we give the derivation of the proposed architecture. We show that the proposed architecture enables us to create the following characteristics simultaneously: (1) high throughput characteristics which are the same as those of the conventional architectures [4]–[6]; (2) a short latency; (3) a good convergence property that is equivalent to that of the LMS; (4) a simple structure which can be implemented with less than half the amount of calculators needed for the conventional architectures [4]–[8]. Note that, in the following, we denote t_{add} as the required time for an addition and t_{mlt} as the required time for a multiplication and we assume the following relation:

$$t_{mlt} \geq t_{add}. \tag{15}$$

3.1 Derivation of the Proposed Architecture

In order to shorten the critical path (CP), the proposed architecture uses binary-tree adders for calculating $y(n)$, and further, it inserts the minimum amount of delay required to obtain the shortest CP ($\mathrm{CP}_{\min}$) of the conventional architectures [4]–[6]. Namely, the $\mathrm{CP}_{\min}$ is given as

$$\mathrm{CP}_{\min} = 2t_{add} + t_{mlt}. \tag{16}$$

Derivation of the proposed architecture is divided in two steps: in Step 1, we explain how to decrease the critical path, and in Step 2, we explain how to cancel the influence of the time lag.

Step 1. Decreasing the Critical path

Let us start by considering the architecture of the LMS. The CP of the LMS ($\mathrm{CP}_{\mathrm{LMS}}$) is the path for calculating the output signal $y(n)$ and its length is described as

$$\mathrm{CP}_{\mathrm{LMS}} = (N+1)t_{add} + 3t_{mlt}. \tag{17}$$

Therefore, it is obvious that $\mathrm{CP}_{\mathrm{LMS}} > \mathrm{CP}_{\min}$.

We decreased the length of the $\mathrm{CP}_{\mathrm{LMS}}$ by using binary-tree adders. Figure 3 (a) shows an 8-tap example of an architecture of the LMS. In this figure, the $\mathrm{CP}_{\mathrm{LMS}}$ is shown by the bold line. By applying a binary-tree adder, we can decrease the length of the CP from $(N+1)t_{add}+3t_{mlt}$ to $(\log_2 N+2)t_{add}+3t_{mlt}$ as is shown in Fig. 3 (b). However, this CP is still longer than the $\mathrm{CP}_{\min}$.

To further decrease the length of the CP, we inserted delays as the conventional methods do. But inserting delays induce the degradation of the convergence characteristics. In the conventional methods, the amount of delay is equal to the filter length. On the other hand, we insert the minimum amount of delay to make the length of the CP equal to that of the $\mathrm{CP}_{\min}$.

We can show how to insert delays using Fig. 4. Figures 4 (a) and 4 (c) show 8-tap and 64-tap examples of the disposition of the calculators in the CP. Note that, we can see from this figure, only three multipliers exist at the beginning and ending of the CP. Hence, by inserting one delay per three calculators, the length of the CP becomes the same as the $\mathrm{CP}_{\min}$ under the assumption made in Eq. (15). Figures 4 (b) and (d) show 8-tap and 64-tap examples of inserting delays and the whole architecture is shown in Fig. 3 (c).

As a result of these insertions, a time lag will be generated whose length is given by

$$D' = \left\lceil \frac{\log_2 N}{3} \right\rceil + 1, \tag{18}$$

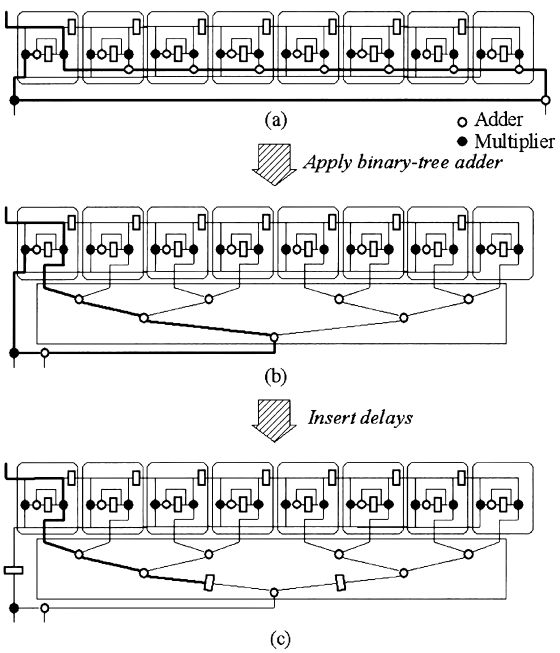

Fig. 3 Modification of SFG (Filter length N = 8-tap). (a) architecture of the LMS, (b) after applying binary-tree adders, (c) after inserting the minimum amount of delay.

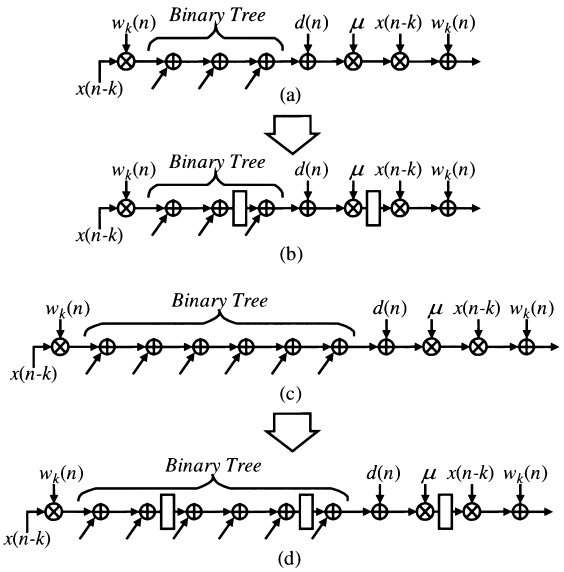

Fig. 4 Example of inserting delays. (a) and (b): $N = 8$-tap, (c) and (d): $N = 64$-tap.

where $\lceil x \rceil$ is the smallest integer larger than x. The equations of the LMS [9] are changed as follows:

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu e(n-D')\mathbf{x}(n-D') \quad (19)$$
$$e(n-D') = d(n-D') - \mathbf{W}^T(n-D')\mathbf{x}(n-D'). \quad (20)$$

Note that Eqs. (19) and (20) correspond to Eqs. (2) and (3) with $D' = D$. This shows that the time lag between $\mathbf{W}(n)$ in Eqs. (19) and (20) becomes D' instead of D. Figure 5 shows the comparison of D' with D. This figure clearly shows that the time lag of the proposed architecture D' become very small compared to that of the conventional architectures D.

Since this time lag causes the degradation of the convergence, we need to cancel the influence of this time lag as the conventional architectures do.

Step 2. Canceling the influence of the time lag

To cancel the influence of this time lag, we add a correctional term $\Lambda'(n)$ to the error signal as the conventional architectures do [4]–[6]. In this case, $\Lambda'(n)$ is defined as

$$\Lambda'(n) = \sum_{i=1}^{D'-1} \mu e(n-2D'-i)\mathbf{x}^T(n-2D'-i) \cdot \mathbf{x}(n-D'). \quad (21)$$

Note the difference between the upper limits of Eqs. (14) and (21); the upper limit changes from D to D'. The required amount of calculation can be reduced by this difference. A comparison with the conventional architectures is given in Sect. 4.

3.2 Formula and Architecture

Finally the formula, on which the proposed architecture is constructed, is described by

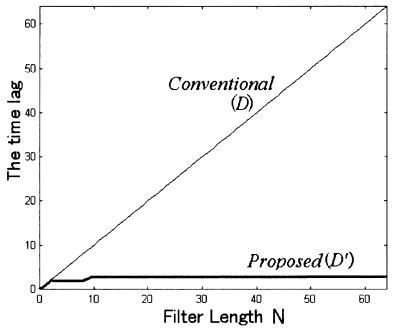

Fig. 5 Comparison of D' with D.

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu\varepsilon(n-D')\mathbf{x}(n-D') \quad (22)$$
$$\varepsilon(n-D') = d(n-D') - \mathbf{W}^T(n-D')\mathbf{x}(n-D') - \Lambda'(n) \quad (23)$$
$$\Lambda'(n) = \sum_{i=1}^{D'-1} \mu e(n-2D'-i)\mathbf{x}^T(n-2D'-i) \cdot \mathbf{x}(n-D'). \quad (24)$$

Figure 6 shows the signal flow graph (SFG) of the proposed architecture: Figure 6 (a) shows the whole architecture, and 6 (b) shows the structure of kth module which corresponds to one tap of the ADF. The structure of $\Lambda'(n)$ is depicted in 6 (c), and 6 (d) shows the structure of a binary-tree adder, where $y_k(n)$ is defined as follows:

$$y_k(n) = x(n-k)w_k(n). \quad (25)$$

The relation between $y(n)$ and $y_k(n)$ are given as

$$y(n) = \sum_{k=0}^{N-1} y_k(n). \quad (26)$$

4. Comparison with the Conventional Architecture

Here, we show the characteristics of the proposed architecture and compare them with those of the conventional ones. In addition, to compare the convergence characteristics of the proposed with the conventional ones, we give some results of computer simulations.

4.1 Comparison of the Characteristics

Figure 7 shows a comparison of the number of the calculators needed for the proposed architecture and the conventional ones; Figures 7 (a), (b) and (c) shows comparisons of their adders, multipliers, and delays. We can see from these figures that the proposed architecture requires less than half the amount of calculations required by the conventional architectures [4]–[8]. Table 1 shows a comparison of the characteristics of the proposed architecture with the conventional ones. Where t_{add} and

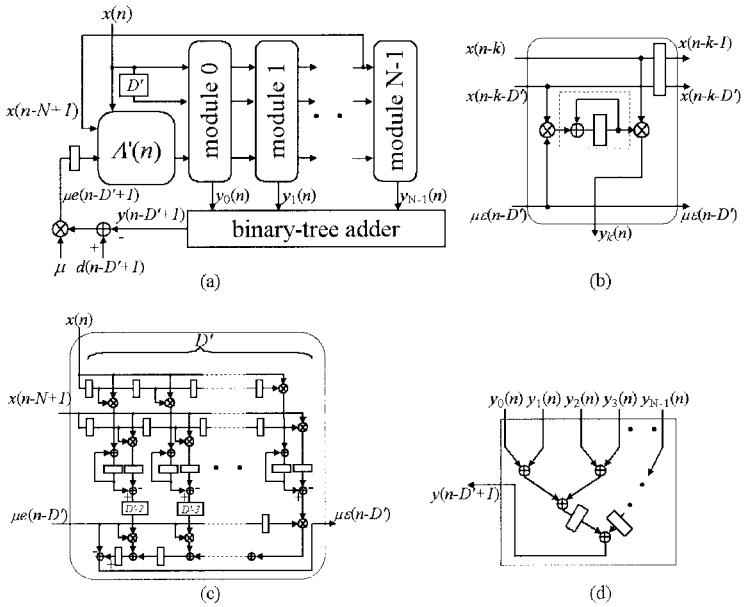

Fig. 6 SFG of the proposed method. (a) whole architecture, (b) module k, (c) correctional term $\Lambda'(n)$, (d) binary-tree adder.

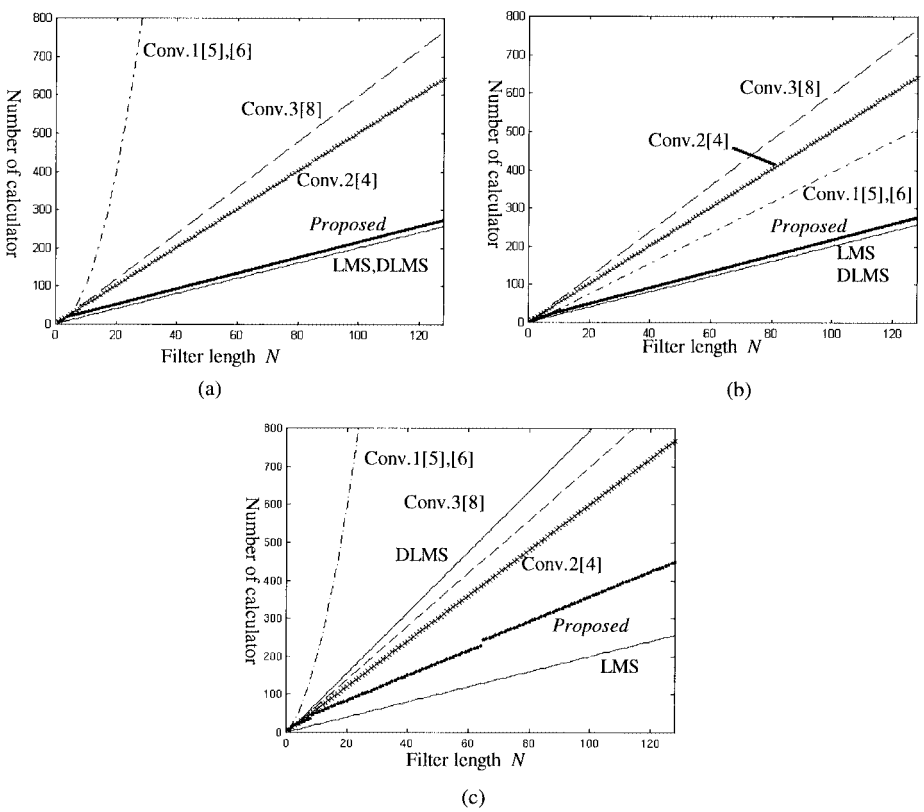

Fig. 7 Comparison of calculators. (a) adder, (b) multiplier, (c) delay.

t_{mlt} are the times necessary to compute a single addition and multiply respectively. We can see, from this table, the proposed architecture has approximately the same latency as that of the LMS.

Table 1 Comparison of the characteristics.

	Critical path	Latency	Number of calculator		
			Adder	Multiplier	Delay
The LMS[9]	$(N+1)t_{add}+3t_{mlt}$	0	$2N+1$	$2N+1$	$2N$
The DLMS[1]-[3]	$2t_{add}+2t_{mlt}$	N	$2N+1$	$2N+1$	$8N-5$
Conv.1(δ=N-3) [5],[6]	$2t_{add}+t_{mlt}$	N	N^2+2	$4N-5$	$N^2+10N-4$
Conv.2[4]	$2t_{add}+t_{mlt}$	N	$5N+1$	$5N+1$	$6N$
Conv.3[8]	$2t_{add}+2t_{mlt}$	0	$5N-2$	$5N-1$	$5N$
Proposed	$2t_{add}+t_{mlt}$	D'-1	$2N+3D'$	$2N+3D'+1$	$3N+(3+D')D'-1+\sum_{k=1}^{D'-1}\left\lceil \frac{N}{2^{k-1}} \right\rceil$

4.2 Comparison of the Throughput

Next, we compare the throughput of the proposed architecture with conventional architectures. Throughput is a reciprocal number of a critical path. Therefore, higher throughput can be obtained as shorter the critical path. It is known that the limiting factor in the throughput of the LMS adaptive filter is the number of additions necessary to form the error signal [9]. Assuming that all of the multiplies in the conventional LMS adaptive filter are computed in parallel, the throughput of the LMS architecture is given by

$$f_{LMS} = \frac{1}{(N+1)t_{add} + 3t_{mlt}}. \tag{27}$$

Unlike the throughput of the LMS architecture, the proposed architecture has a throughput that is independent of its filter length. The throughput of the proposed architecture is given by

$$f_{prop} = \frac{1}{2t_{add} + t_{mlt}}. \tag{28}$$

Therefore it can be implemented without a significant decrease in the sample rate of the system.

4.3 Simulation Results

To verify the validity of the proposed architecture, we show some of the results of the computer simulation of system identification. The optimum system was a low-pass finite impulse response (FIR) filter of length 10. The length of the ADF was selected to be the same as that of the optimum system. The amount of delay D was 10 for the conventionals and $D' = 3$ for the proposed. The input signal $\mathbf{x}(n)$ was a white Gaussian process with mean = 0 and variance = 1. The step size parameter μ was selected to be the maximum value that guarantees the convergence of the mean-squared error (MSE) [9]. For a comparison reference, we used the impulse response error ratio (IRER).

Figure 8 shows the learning curves of the proposed architecture and of the conventional ones, where Conv.1 is for the LDLMS [5], [6], Conv.2 is for the algorithm proposed in [4], and Conv.3 is for the algorithm proposed in [8]. Note that Conv.1 has a parameter δ and here we choose $\delta = 7$ [5], [6]. From the figure we can verify that the learning curve of the proposed architecture is equivalent to that of the LMS.

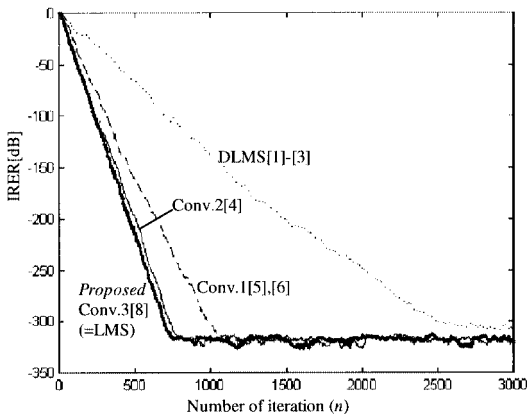

Fig. 8 Comparison of learning curves of the proposed architecture with that of the conventional methods.

5. Conclusion

In this paper we proposed a new pipelined architecture for the DLMS algorithm. This architecture enabled us to simultaneously achieve good convergence characteristics, a short latency, and high throughput characteristics with less than half the amount of calculation used in the conventional architectures. To verify the validity of the proposed architecture, we showed some of the results of the computer simulation of system identification and compared the characteristics of the proposed architecture with the conventional ones.

References

[1] R. Haimi-Cohen, H. Herzberg, and Y. Be'ery, "Delayed adaptive LMS filtering: Current results," Proc. IEEE ICASSP '90, pp.1273–1276, April 1990.
[2] G. Long, F. Ling, and J.G. Proakis, "The LMS algorithm with delayed coefficient adaptation," IEEE Trans. Acoust., Speech & Signal Process., vol.37, no.9, pp.1397–1405, Sept. 1989.
[3] G. Long, F. Ling, and J.G. Proakis, "Corrections to "The LMS algorithm with delayed coefficient adaptation"," IEEE Trans. Acoust., Speech & Signal Process., vol.40, no.1, pp.230–232, Sept. 1989.
[4] S.C. Douglas, Q. Zhu, and K.F. Smith, "A pipelined LMS adaptive FIR filter architecture without adaptation delay,"

IEEE Trans. Signal Processing, vol.46, no.3, pp.775–779, March 1998.
[5] K. Matsubara, K. Nishikawa, and H. Kiya, "A new pipelined architecture of the LMS algorithm without degradation of convergence characteristics," Proc. IEEE ICASSP '97, pp.4125–4128, April 1997.
[6] K. Matsubara, K. Nishikawa, and H. Kiya, "Pipelined LMS adaptive filter using a new look-ahead transformation," Proc. IEEE ISCAS '97, pp.2309–2312, June 1997.
[7] R.D. Poltmann, "Conversion of the delayed LMS algorithm into the LMS algorithm," IEEE Signal Proc. Letters, vol.2, no.12, p.223, Dec. 1995.
[8] A. Harada, K. Nishikawa, and H. Kiya, "Pipelined architecture of the LMS adaptive digital filter with the minimum output latency," IEICE Trans. Fundamentals, vol.E81-A, no.8, pp.1578–1585, Aug. 1998.
[9] N.R. Shanbhag and K.K. Parhi, Pipelined Adaptive Digital Filters, Kluwer, Boston, MA, 1994.

Tadaaki Kimijima was born in Osaka, Japan, on September 1, 1974. He received the B.E. degrees in electronics and information engineering from Tokyo Metropolitan University in 1997. He is currently a candidate for the M.E. degree at Tokyo Metropolitan University. His research interest is in adaptive signal processing.

Kiyoshi Nishikawa was born in Tokyo, Japan, on December 12, 1966. He received the B.E., the M.E., and the D.E. degrees in electrical engineering from Tokyo Metropolitan University in 1990, 1992 and 1996, respectively. From 1992 to 1993, he was at the Computer Systems Laboratory, Nippon Steel Corp. as a researcher. Since 1993, he has been with Tokyo Metropolitan University as a research associate. His research interest includes the adaptive signal processing. He is a member of IEEE SP, CAS, COM, and Computer Societies.

Hitoshi Kiya was born in Yamagata, Japan, on November 16, 1957. He received the B.E. and M.E. degrees in electrical engineering from Nagaoka University of Technology, Niigata, Japan, and the D.E. degree in electrical engineering from Tokyo Metropolitan University, Tokyo, Japan, in 1980, 1982, and 1987, respectively. In 1982, he joined Tokyo Metropolitan University, where he is currently an Associate Professor of Electrical Engineering, Graduate School of Engineering. He was a visiting researcher of the University of Sydney in Australia from Oct. 1995 to Mar. 1996. His research interests are in digital signal processing, multirate systems, adaptive filtering, image processing, and efficient algorithms for VLSI implementation. Dr. Kiya is a Member of the IEEE, the Image Electronics Engineers of Japan and the Institute of Television Engineers of Japan. He is an Associate Editor of IEEE Transactions on Signal Processing of IEEE.